# Proje 1: Veritabanı Performans Optimizasyonu ve İzleme

## 1. Giriş

Günümüzde veritabanı sistemleri, kurumsal bilgi yönetiminin temel taşlarından biri haline gelmiştir. Artan veri miktarı ve karmaşık sorgular, veritabanı performansını doğrudan etkilemekte ve sistemlerde yavaşlamalara yol açabilmektedir. Bu bağlamda, performans izleme, analiz ve iyileştirme çalışmaları; sistem verimliliğini artırmak, kullanıcı deneyimini geliştirmek ve kaynak kullanımını optimize etmek adına kritik öneme sahiptir.

Bu proje kapsamında Microsoft SQL Server ortamında, büyük ve ilişkisel bir veritabanı üzerinde performans analizi gerçekleştirilmiş; sorgu iyileştirmeleri, indeks yönetimi, veri erişim rolleri gibi çeşitli teknikler uygulanarak performans optimizasyonuna yönelik adımlar atılmıştır. Proje süresince yapılan tüm işlemler hem teorik hem de pratik açıdan belgelendirilmiş ve sonuçları değerlendirilmiştir.

**1.1. Projenin Amacı**

Bu projenin temel amacı, bir veritabanı sisteminde performans sorunlarını tespit etmek ve bu sorunlara çözüm olacak teknikleri uygulayarak sistemin genel verimliliğini artırmaktır. Proje çerçevesinde şu alt hedefler belirlenmiştir:

* **Yavaş çalışan sorguların tespiti** ve bu sorguların nedenlerinin analiz edilmesi
* **SQL Profiler ve DMV (Dynamic Management Views)** gibi araçlar kullanarak sorgu performansının izlenmesi
* **Eksik veya gereksiz indekslerin analiz edilmesi**, uygun indeksleme stratejilerinin uygulanması
* **Performansı düşük sorguların optimize edilmesi** ve daha etkili sorgu yapılarının kullanılması
* **Veri yöneticisi rolleri ve yetkilendirme politikaları** üzerinden erişim kontrolü sağlanarak güvenliğin artırılması

Bu amaçlara ulaşmak, gerçek dünyadaki veritabanı yöneticilerinin karşılaştığı tipik performans problemlerine çözüm üretme becerisini geliştirmeyi hedeflemektedir.

**1.2. Kullanılan Veritabanı ve Araçlar**

Bu çalışmada, Microsoft tarafından sağlanan **AdventureWorks2022** örnek veritabanı kullanılmıştır. AdventureWorks, geniş veri tabanlı yapısıyla satış, ürün, müşteri, stok ve personel gibi birçok ilişkilendirilmiş veri tablosunu içermektedir. Bu yapı, gerçek dünya veritabanı yapısına oldukça yakındır ve performans testleri için uygun bir ortam sunar.

Kullanılan temel araçlar ve teknolojiler şunlardır:

* **Microsoft SQL Server Management Studio (SSMS):** Tüm SQL sorgularının yazıldığı ve çalıştırıldığı ortam.
* **SQL Server Profiler:** Gerçek zamanlı sorgu izleme ve performans ölçüm aracı.
* **Dynamic Management Views (DMV):** Sistem seviyesinde performans istatistikleri ve kullanım bilgilerini almak için kullanılan dinamik görünümler.
* **T-SQL (Transact-SQL):** MSSQL’e özel sorgulama dili ile indeks, sorgu ve yetki işlemleri gerçekleştirilmiştir.

Tüm işlemler MSSQL 2019 Express Edition üzerinde, tek kullanıcı oturumunda yürütülmüş ve sonuçlar ekran görüntüleriyle belgelenmiştir.

## 2. Veritabanı İzleme ve Analiz

Veritabanı performansının değerlendirilmesinde ilk adım, sistemde çalışan sorguların ve işlemlerin izlenmesidir. Bu süreçte hem **SQL Server Profiler** hem de **Dynamic Management Views (DMV)** kullanılarak sistemdeki yavaş sorgular ve darboğazlar belirlenmiştir. Aşağıda her iki yöntemin nasıl uygulandığı detaylı bir şekilde açıklanmıştır.

**2.1. SQL Server Profiler ile İzleme**

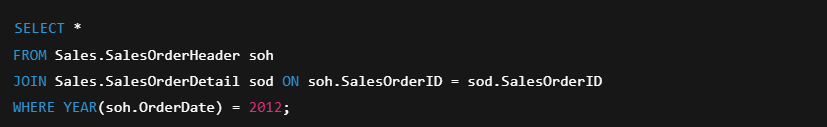
**SQL Server Profiler**, Microsoft SQL Server ile birlikte gelen güçlü bir araçtır. Gerçek zamanlı olarak çalışan SQL komutlarını, bekleme sürelerini, CPU kullanımını ve IO istatistiklerini izlemeye olanak sağlar.

**Uygulama Adımları:**

1. **SQL Server Profiler** aracı başlatıldı.
2. File > New Trace seçeneği ile yeni bir izleme başlatıldı.
3. Bağlantı sağlandıktan sonra:
   * **Template olarak "TSQL\_Duration"** seçildi.
   * Events Selection sekmesinde Duration kolonuna filtre uygulandı:

Bu sayede sadece 1 saniyeden uzun süren sorgular izlenmiştir.

1. İzleme başlatıldı ve AdventureWorks2022 veritabanı üzerinde aşağıdaki gibi örnek yavaş sorgular çalıştırıldı:



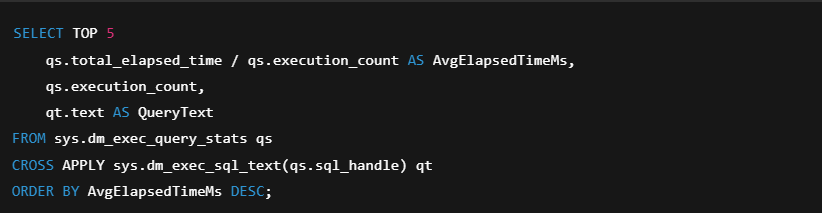
**Sonuç:**

Bu yöntemle yavaş çalışan sorgular belirlenmiş ve sonraki adım olan sorgu iyileştirme süreci için temel oluşturulmuştur.

**2.2. Dynamic Management Views (DMV) ile Sorgu Performans Analizi**

**DMV (Dynamic Management Views)**, SQL Server'ın çalışma zamanında sistem durumu, sorgu istatistikleri, bellek ve kaynak kullanımı gibi bilgileri sağlayan sistem görünümüdür. Bu proje kapsamında sys.dm\_exec\_query\_stats görünümü kullanılarak geçmişte en uzun süren sorgular tespit edilmiştir.

**Kullanılan Sorgu:**

 **Açıklama:**

* total\_elapsed\_time: Toplam çalışma süresi
* execution\_count: Kaç kere çalıştırıldığı
* AvgElapsedTimeMs: Ortalama çalışma süresi
* QueryText: Çalışan sorgunun kendisi

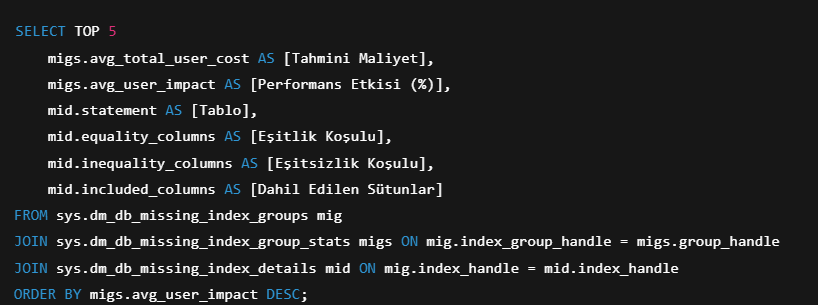
## 3. İndeks Yönetimi

İndeksler, veritabanı tablolarında sorguların daha hızlı çalışmasını sağlayan veri yapılarıdır. Özellikle büyük veri kümelerinde doğru indekslerin oluşturulması, sistem performansını doğrudan artırır. Bu projede hem eksik hem de gereksiz indeksler analiz edilmiş ve gerekli optimizasyon işlemleri uygulanmıştır.

**3.1. Eksik İndekslerin Tespiti**

SQL Server, çalıştırılan sorgulara dayanarak bazı indeks önerilerinde bulunabilir. Bu bilgiler sys.dm\_db\_missing\_index\_details, sys.dm\_db\_missing\_index\_group\_stats ve sys.dm\_db\_missing\_index\_groups gibi DMV’lerde tutulur.

**Kullanılan Sorgu:**

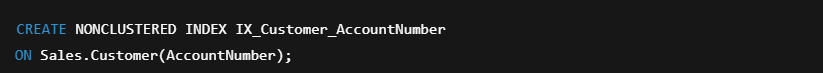
 **Açıklama:**

* avg\_user\_impact: Bu indeksin sorgu performansına % kaç katkı sağlayacağı
* statement: İlgili tablo
* equality\_columns: WHERE veya JOIN'de kullanılan eşitlik sütunları
* included\_columns: Seçilen sütunlar (SELECT)

**3.2. Yeni İndeks Oluşturma**

Eksik indeks analizine dayanarak örnek bir tablo üzerinde performansı artıracak bir indeks oluşturulmuştur. Örnek olarak Customer tablosu üzerinde bir arama kriteri olan AccountNumber sütununa indeks eklenmiştir.

**Kullanılan Sorgu:**

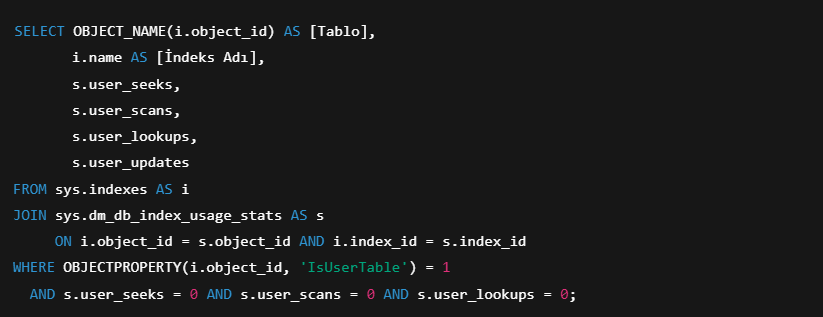
 **Açıklama:**

* NONCLUSTERED: Tablo yapısını etkilemeden indeks oluşturur
* IX\_...: Standart indeks isimlendirme konvansiyonu

**3.3. Gereksiz İndekslerin Tespiti**

Veritabanında oluşturulmuş ancak sistem tarafından nadiren ya da hiç kullanılmayan indeksler, hem disk alanı israfına hem de yazma işlemlerinde yavaşlamaya neden olur. Bu nedenle kullanılmayan indekslerin tespiti de performans iyileştirmesi açısından önemlidir.

**Kullanılan Sorgu:**

 **Açıklama:**

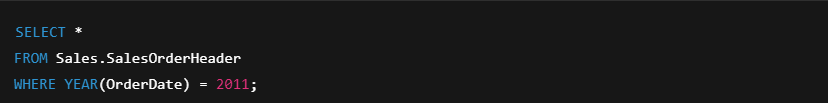
* user\_seeks, user\_scans, user\_lookups: İndeksin ne kadar kullanıldığını gösterir
* 0 olanlar, sistem tarafından hiç kullanılmamış indekslerdir

## 4. Sorgu İyileştirme

Veritabanı performansını etkileyen en önemli faktörlerden biri sorguların yazım şeklidir. Kötü yazılmış, filtreleme yapmayan, tüm tabloyu tarayan veya uygun indeks kullanmayan sorgular ciddi performans sorunlarına yol açabilir. Bu bölümde, önce kötü bir sorgu analiz edilmiş, ardından daha verimli hale getirilmiş optimize edilmiş versiyonu oluşturulmuştur.

**4.1. Kötü Performanslı Sorgu Analizi**

**Problemli Sorgu:**

 **Açıklama:**

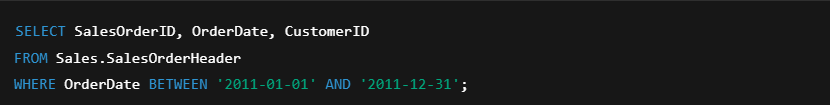
* Bu sorgu \* ile tüm sütunları seçmektedir. Bu, gereksiz veri transferine sebep olur.
* YEAR(OrderDate) fonksiyonu, OrderDate sütununun tamamı üzerinde işlem yaptığı için indeks kullanılamaz ve tablo taraması yapılır (**index scan**).
* WHERE filtresi işlevsel olarak doğrudur fakat performans açısından olumsuzdur.

**Sonuç:**

* SQL Profiler veya DMV üzerinden izlendiğinde bu sorgunun **yüksek CPU ve IO** kullandığı, uzun süre çalıştığı gözlemlenmiştir.
* **Execution Plan** (Yürütme Planı) incelendiğinde, indekslerin devre dışı kaldığı görülmüştür.

**4.2. Sorgunun Optimize Edilmiş Hali**

**İyileştirilmiş Sorgu:**

 **Neden Daha İyi?**

* \* yerine sadece gerekli sütunlar seçilmiştir (veri transferi azaltılır).
* YEAR(OrderDate) fonksiyonu kaldırılmış, bunun yerine **BETWEEN** operatörü kullanılmıştır.
* Bu yapı sayesinde OrderDate sütunu üzerinde varsa indeks kullanılabilir (**index seek** gerçekleşir).
* Bu da SQL Server’ın tabloyu baştan sona taramak yerine sadece ilgili aralığı işlemesine olanak tanır.

**Karşılaştırma:**

| **Kriter** | **Kötü Sorgu** | **İyi Sorgu** |
| --- | --- | --- |
| Fonksiyon Kullanımı | YEAR(OrderDate) → Index devre dışı | BETWEEN → Index kullanılabilir |
| Seçilen Sütunlar | \* → Tüm tablo | Yalnızca gerekli sütunlar |
| Performans | Yavaş, tüm tablo taranıyor | Hızlı, indeks kullanımı mümkün |

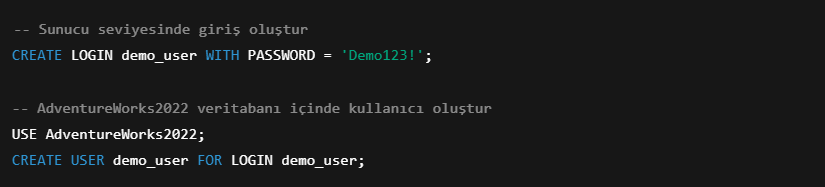
## 5. Veri Yöneticisi Rolleri ve Yetkilendirme

Veritabanı yönetiminde yalnızca performans değil, **güvenlik ve erişim kontrolü** de son derece önemlidir. Yetkisiz erişimlerin önlenmesi, sadece gerekli kişilere gerekli yetkilerin verilmesi gibi uygulamalar veri bütünlüğünü ve gizliliğini sağlar. Bu bölümde SQL Server’da kullanıcı oluşturma, yetki atama ve yetkiyi kaldırma işlemleri gerçekleştirilmiştir.

**5.1. Kullanıcı Oluşturma**

Veritabanına dışarıdan bağlanacak bir kullanıcı için önce bir **login (giriş hesabı)** oluşturulmalı, ardından bu login'e özel bir **veritabanı kullanıcısı (user)** tanımlanmalıdır.

**Kullanılan Komutlar:**

 **Açıklama:**

* CREATE LOGIN: SQL Server'a giriş yapabilen genel bir hesap oluşturur.
* CREATE USER: Veritabanına özgü kullanıcıyı tanımlar.
* Şifre politikalarına uygun, karmaşık bir şifre kullanılmalıdır (Demo123! örneği gibi).

**5.2. Yetki Verme ve Geri Alma**

Demo kullanıcı oluşturulduktan sonra bu kullanıcıya belirli tablolara erişim yetkisi verilir. Daha sonra yetkisi kaldırılarak kontrol tamamlanır.

**Yetki Verme (GRANT)**

 **Açıklama:**

* Bu komut, demo\_user kullanıcısına sadece SalesOrderHeader tablosu üzerinde **SELECT** (okuma) yetkisi tanır.
* Kullanıcı bu tabloya erişebilir ancak veri değiştiremez.

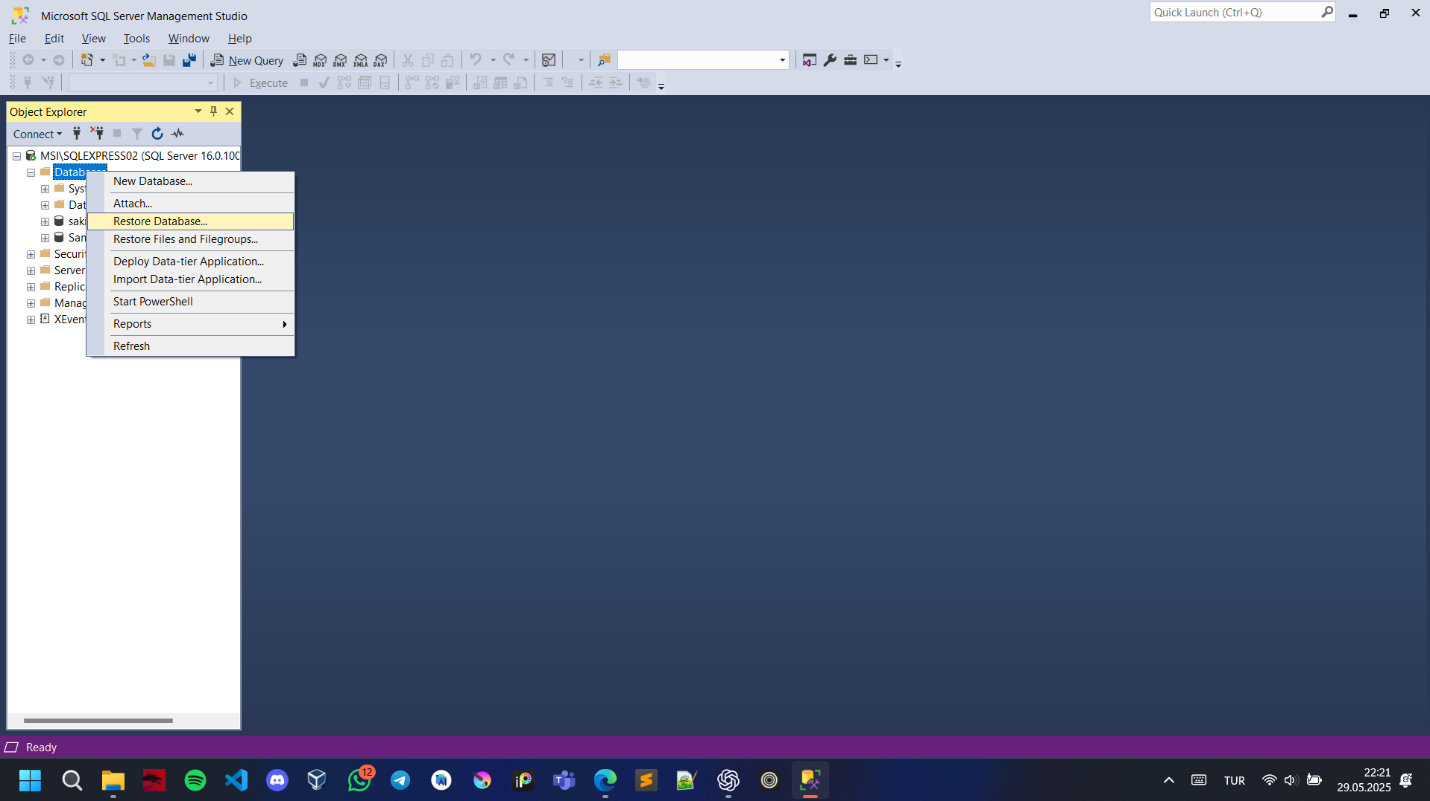
**Yetki Kaldırma (REVOKE)**

 **Açıklama:**

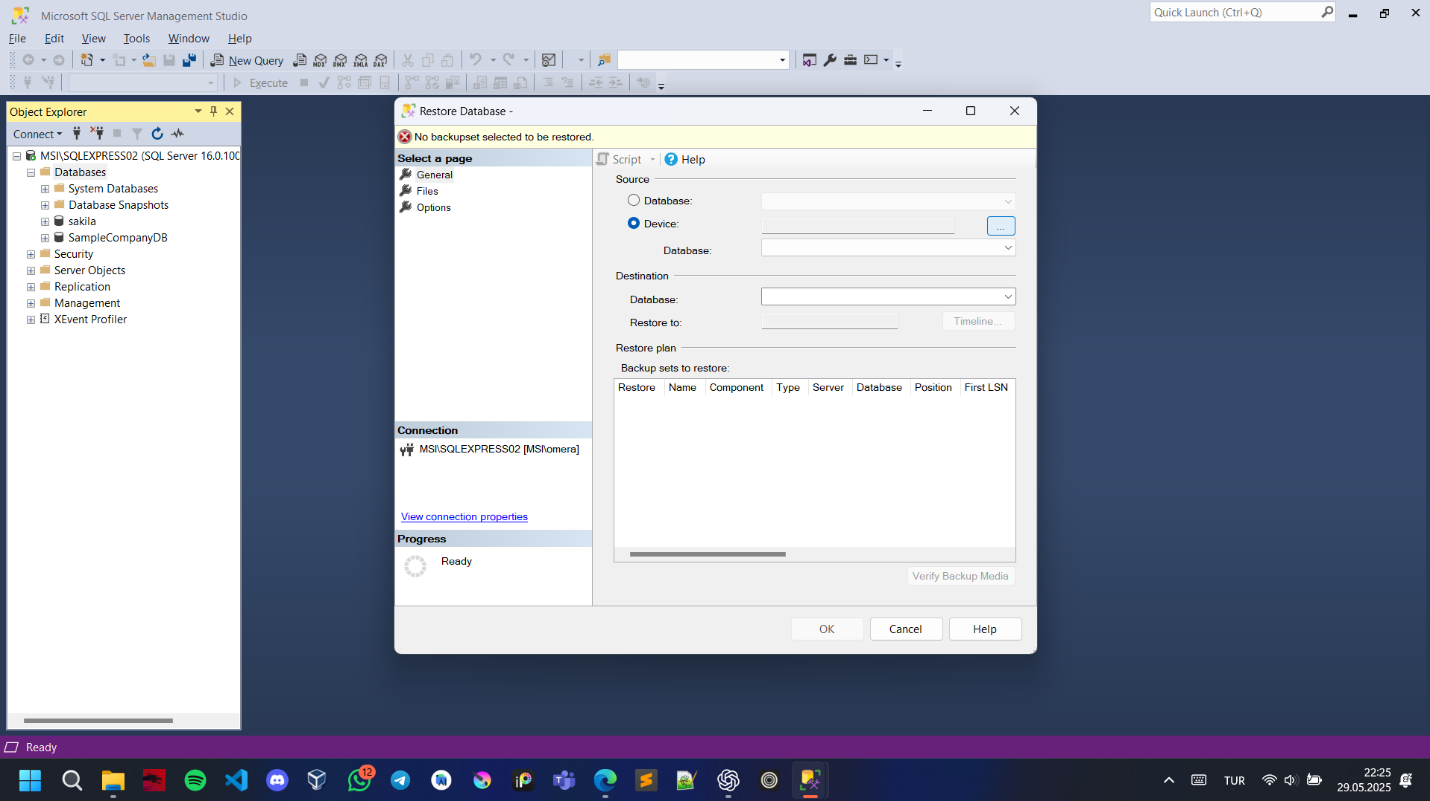
* Bu komutla demo\_user kullanıcısının okuma yetkisi geri alınır.
* Artık bu tabloya erişim sağlayamaz.

## 6.Ekran Görüntüleri

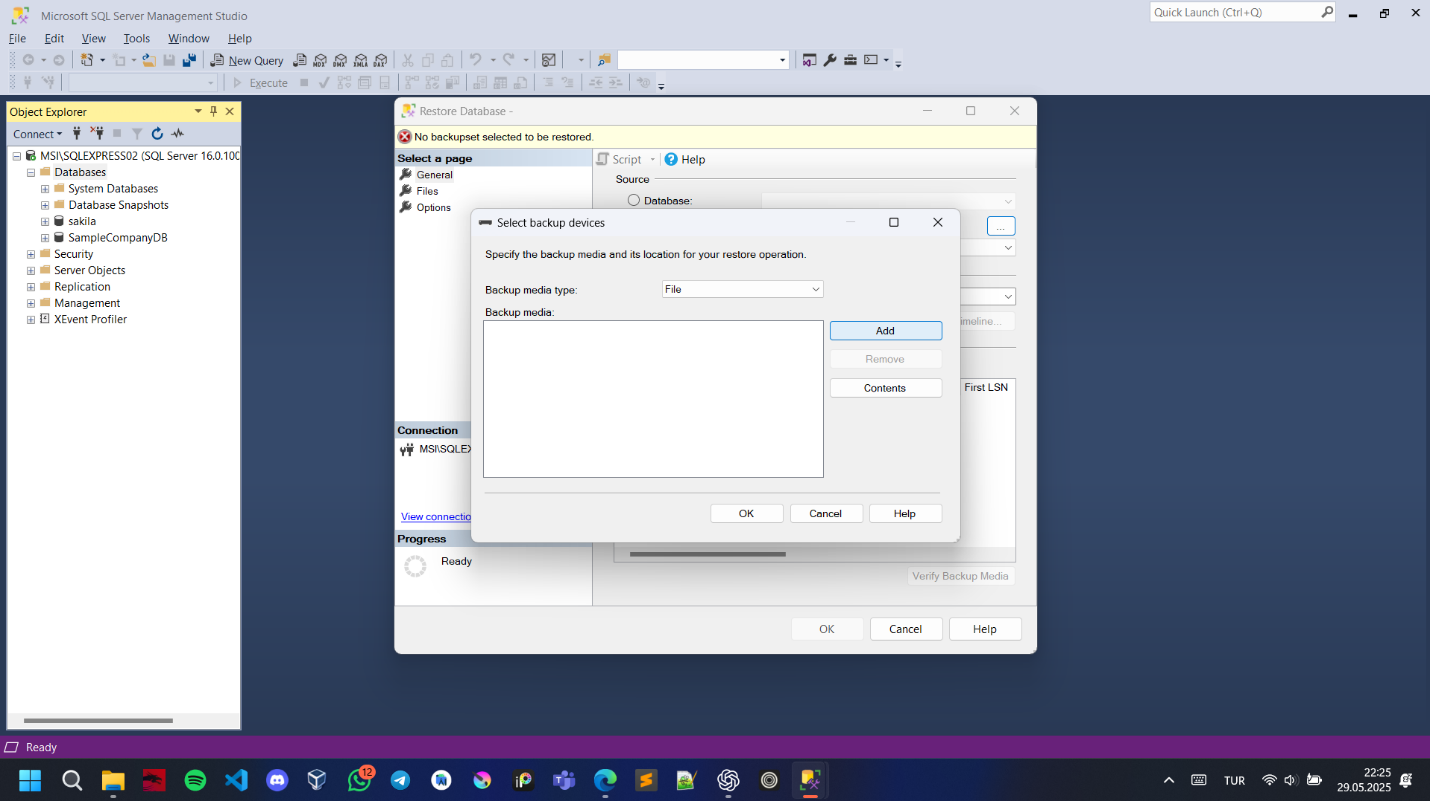
1)



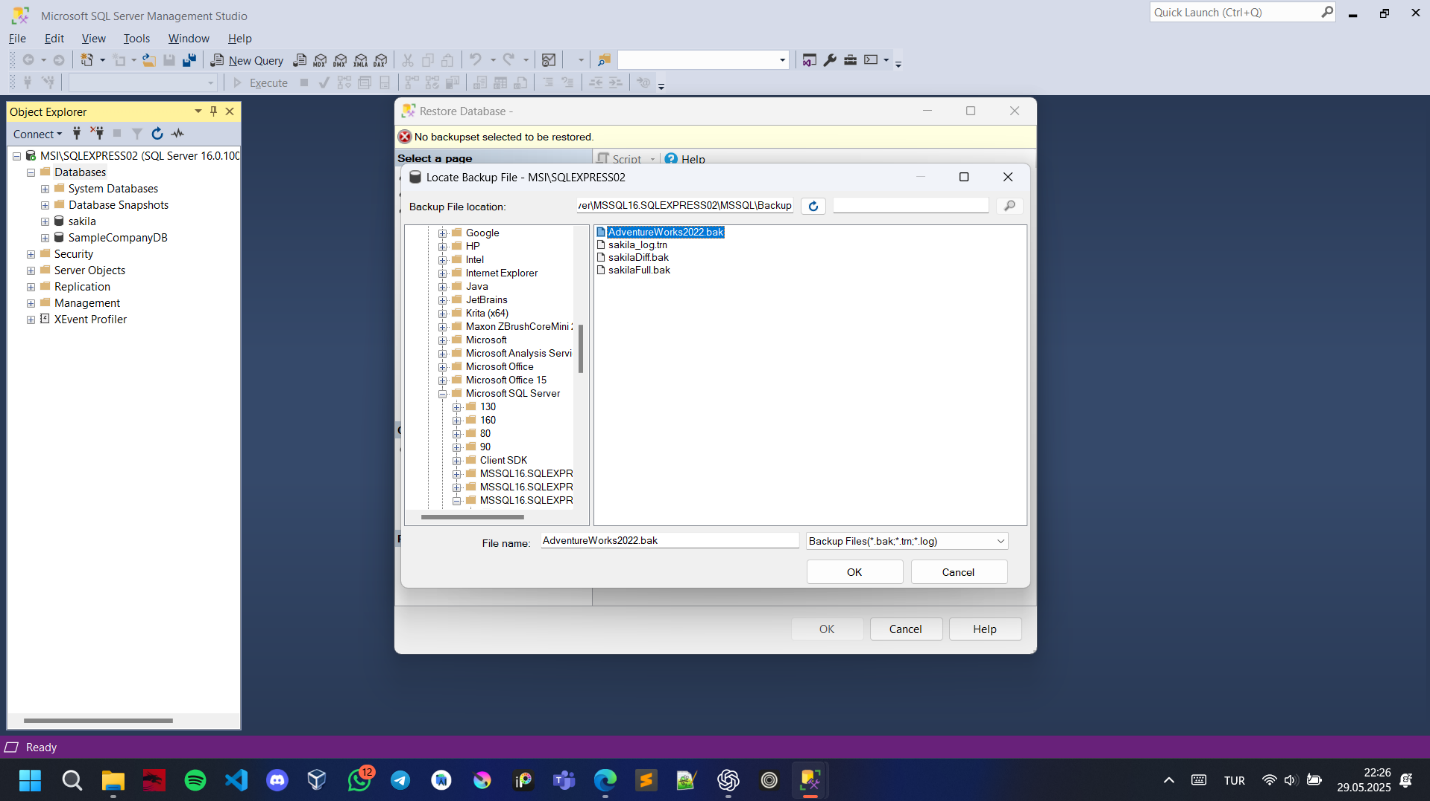
2)



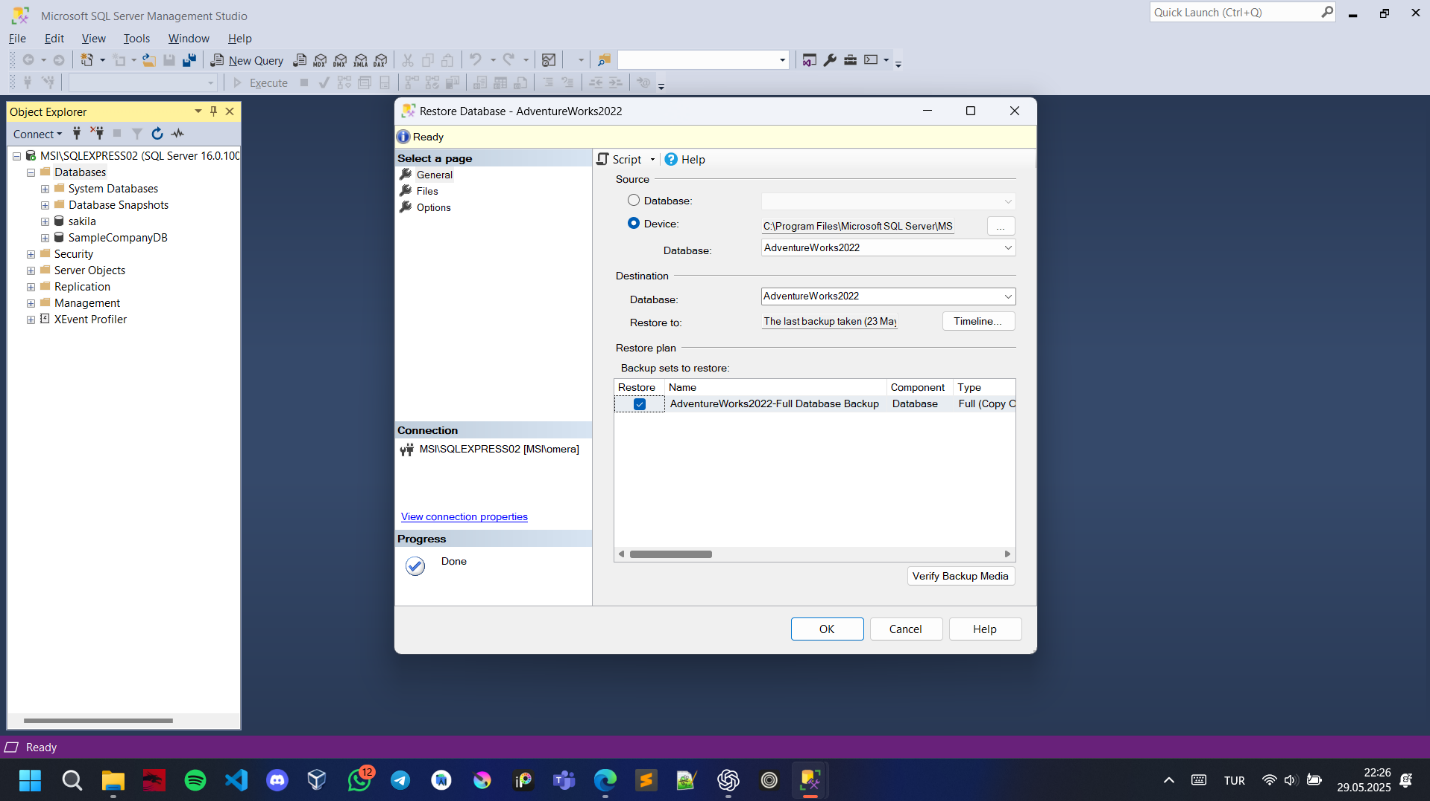
3)



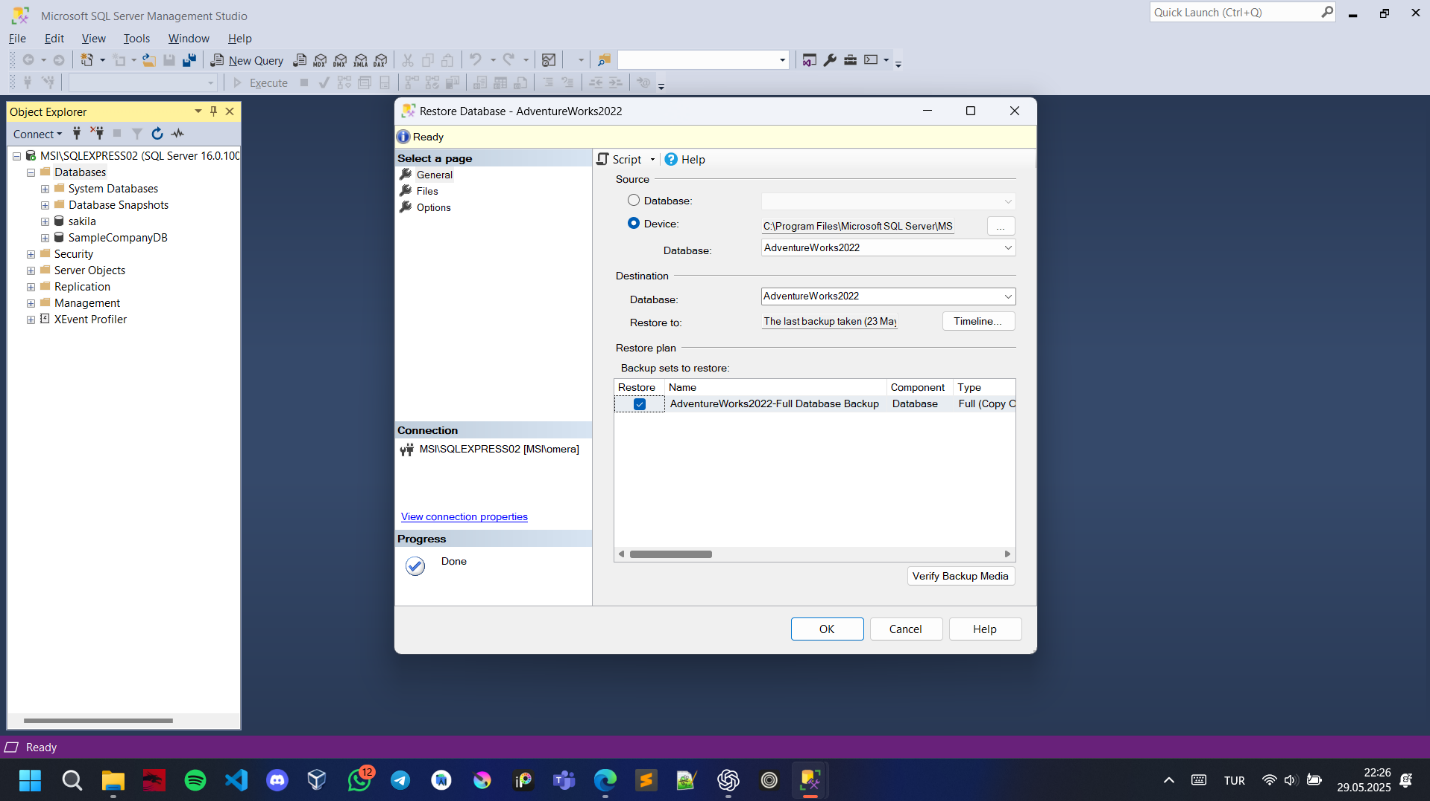
4)



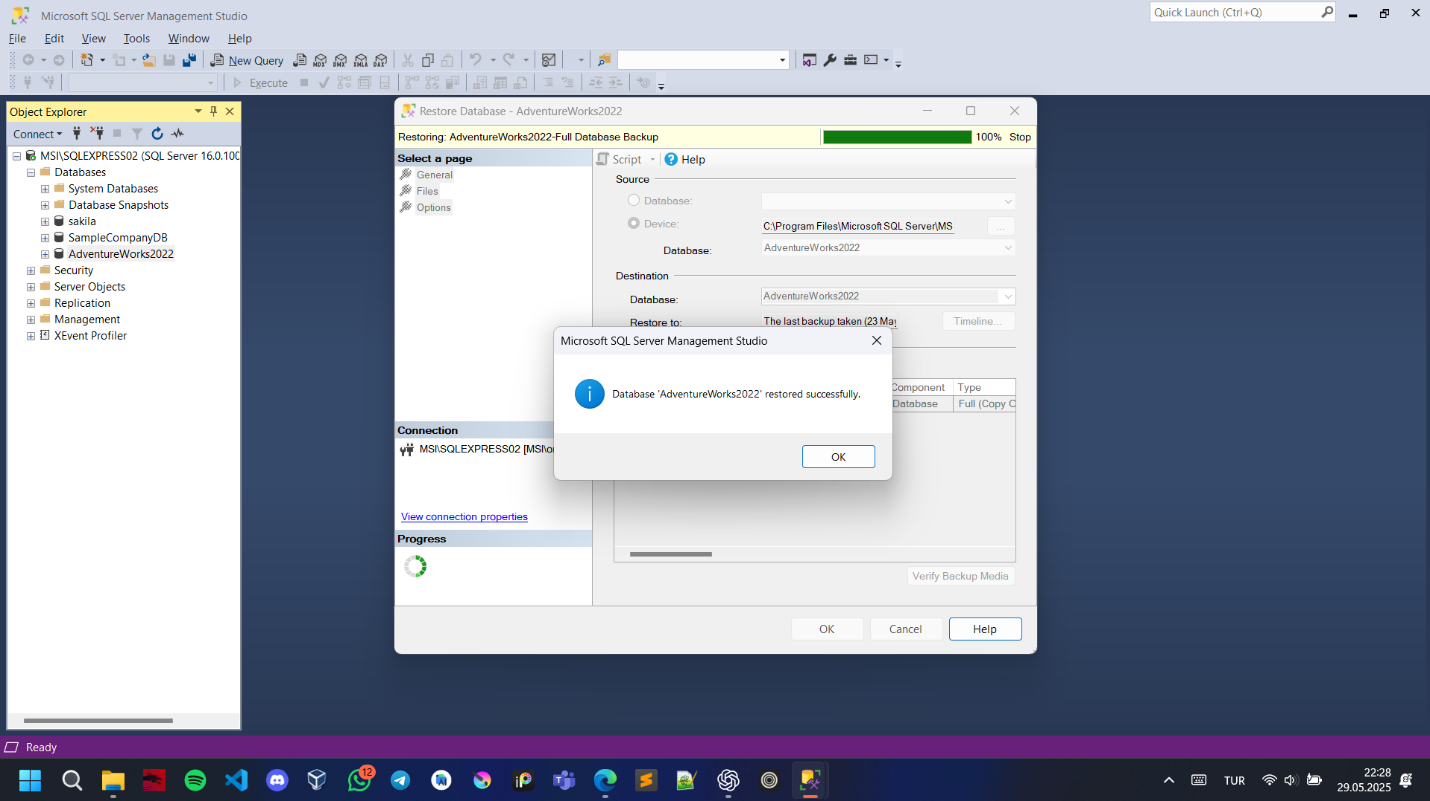
5)



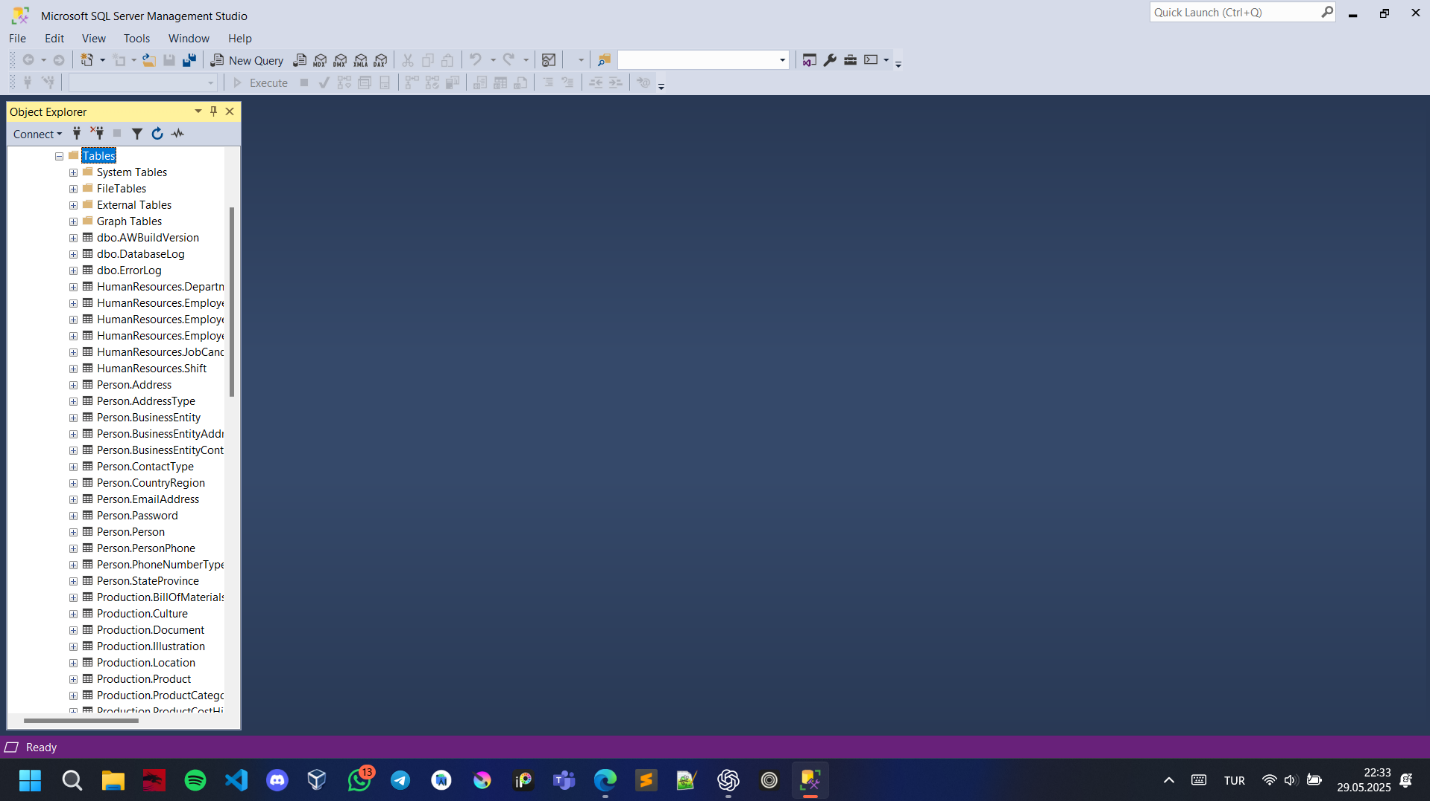
6)



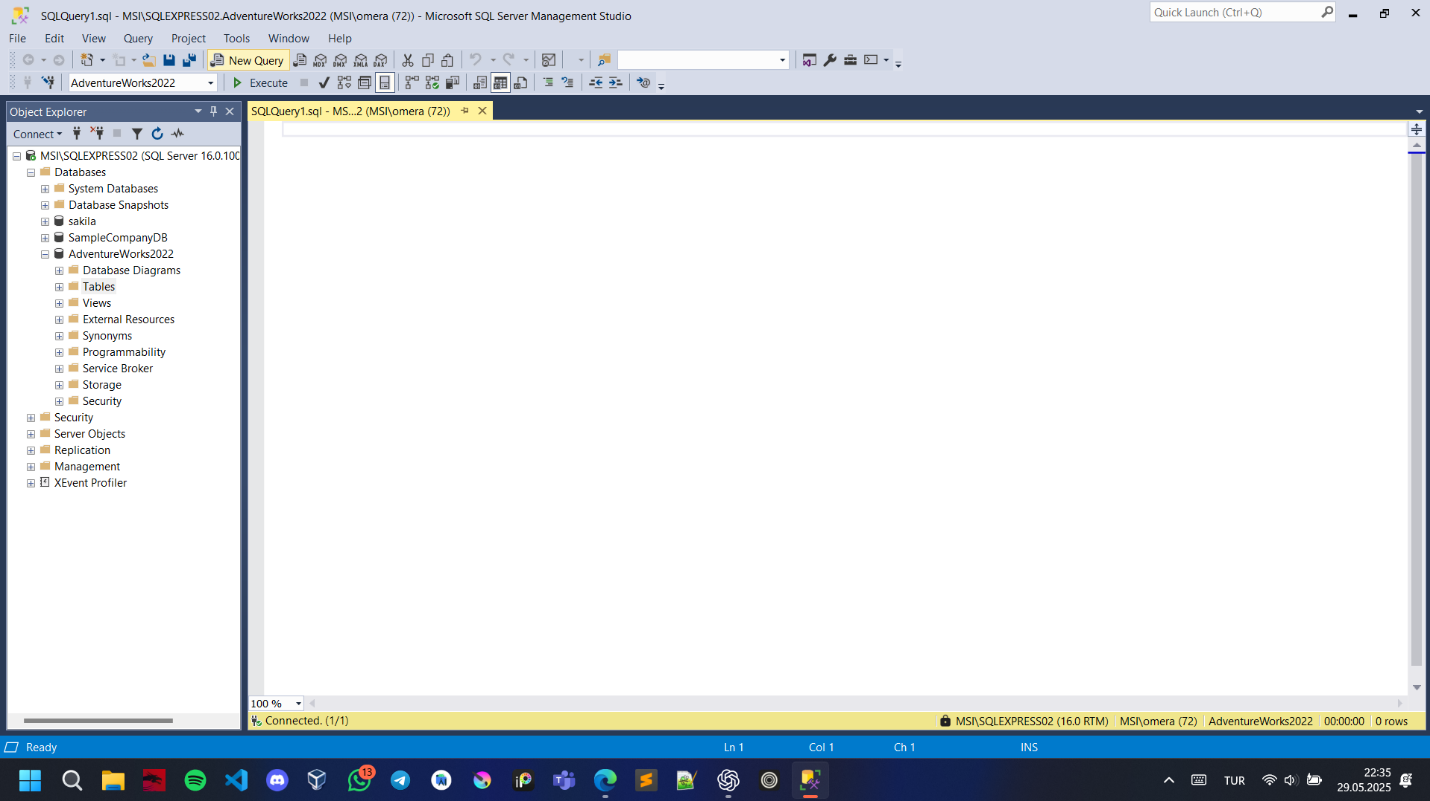
7)



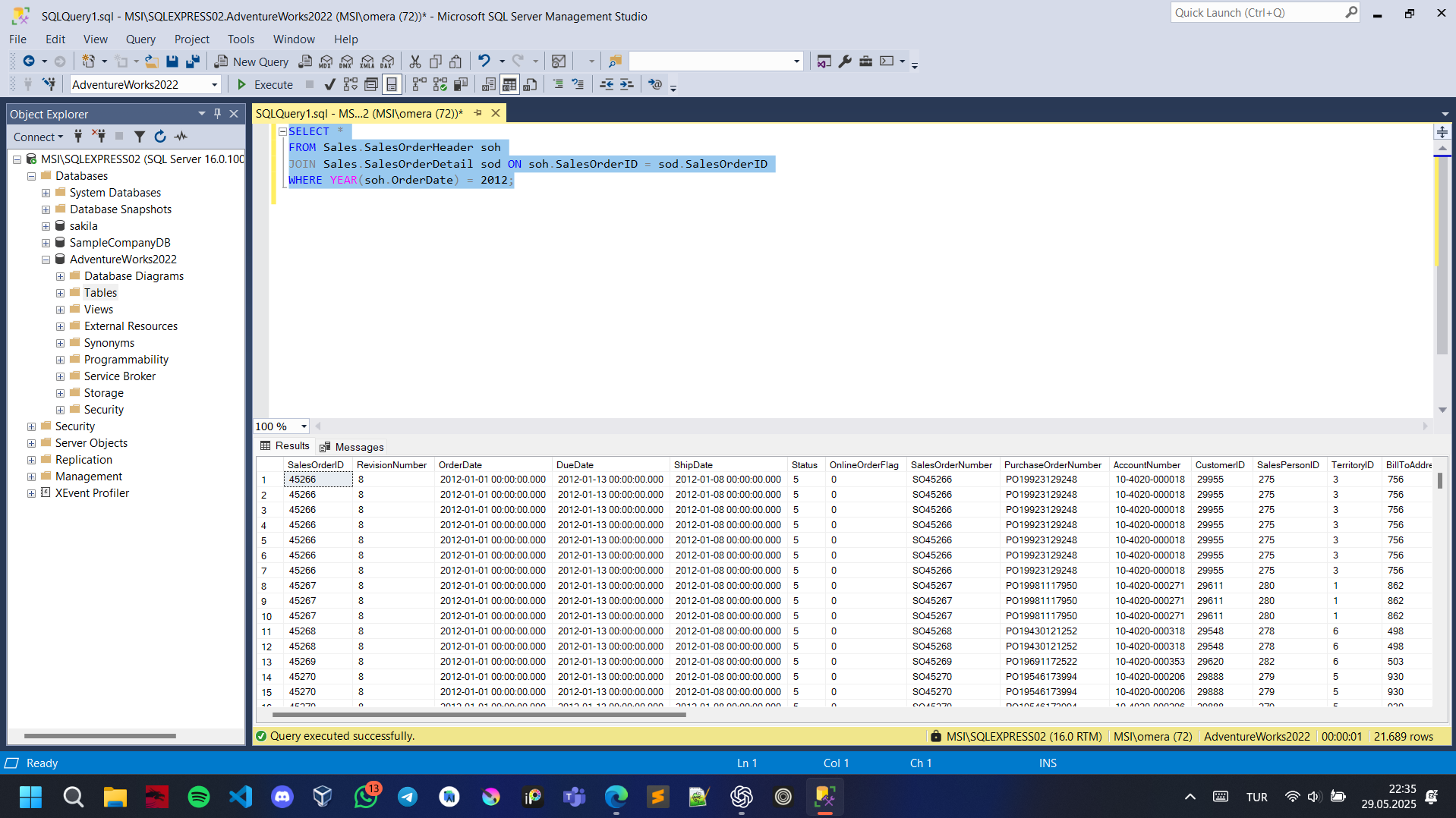
8)



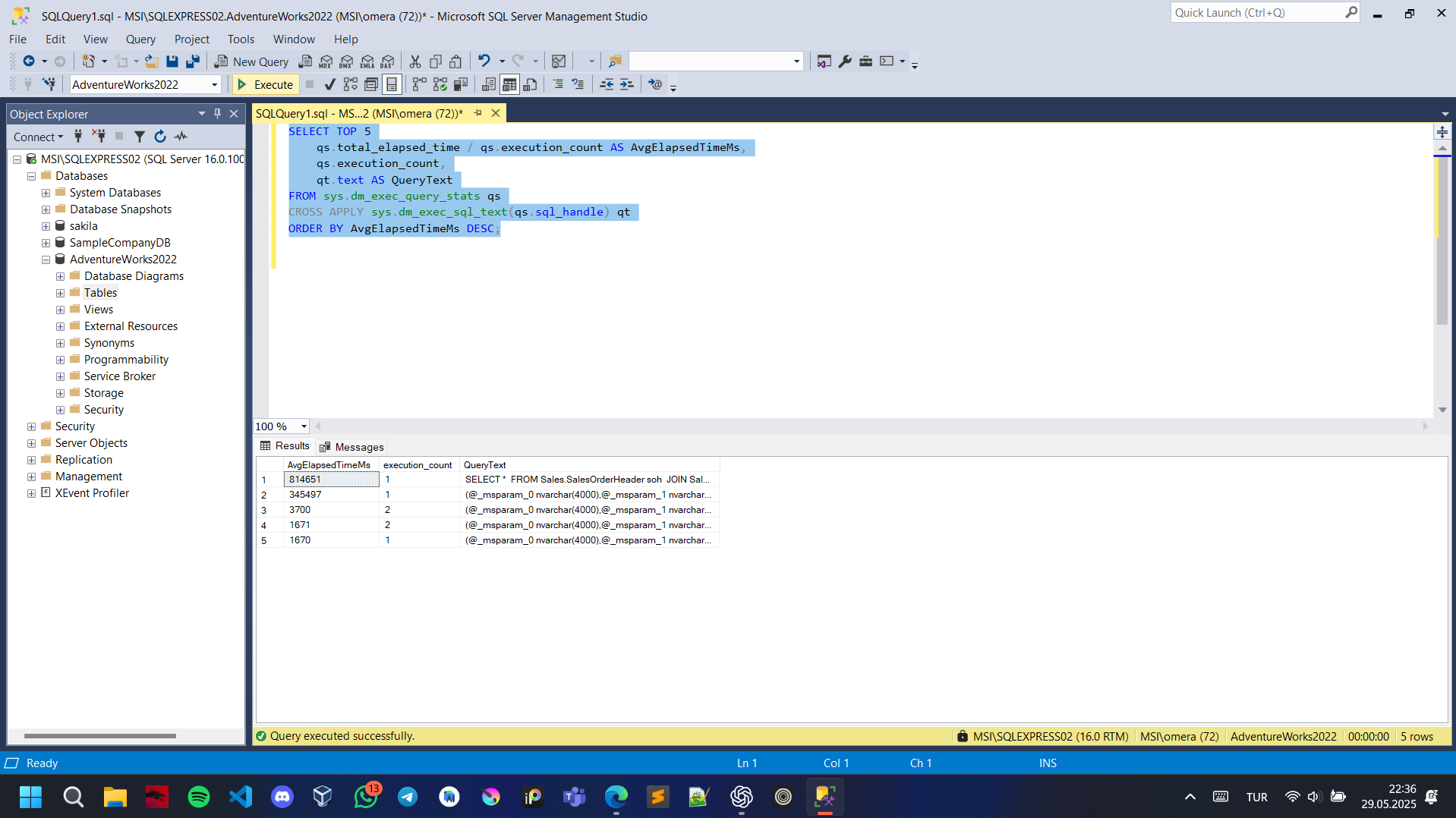
9)



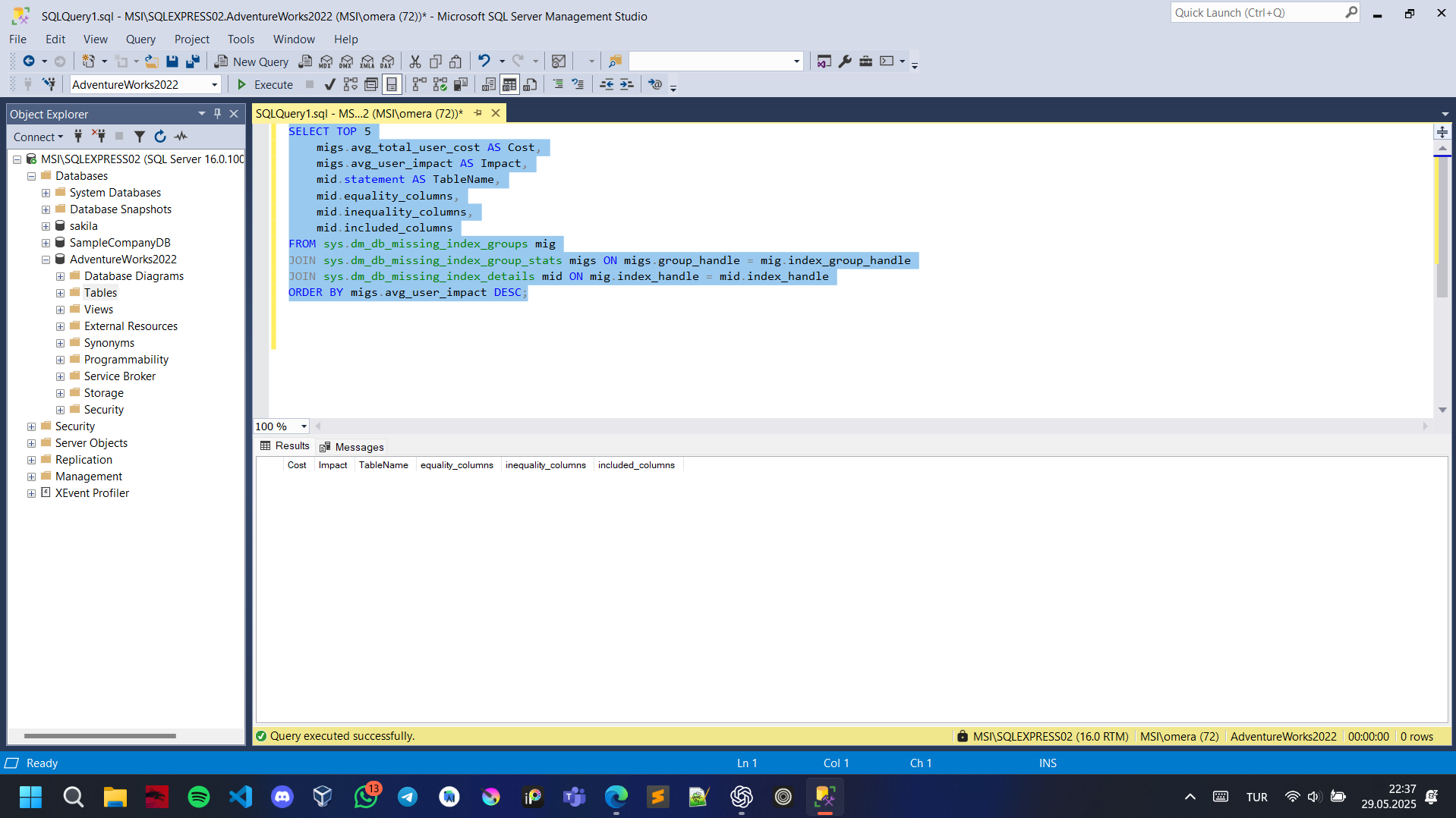
10)



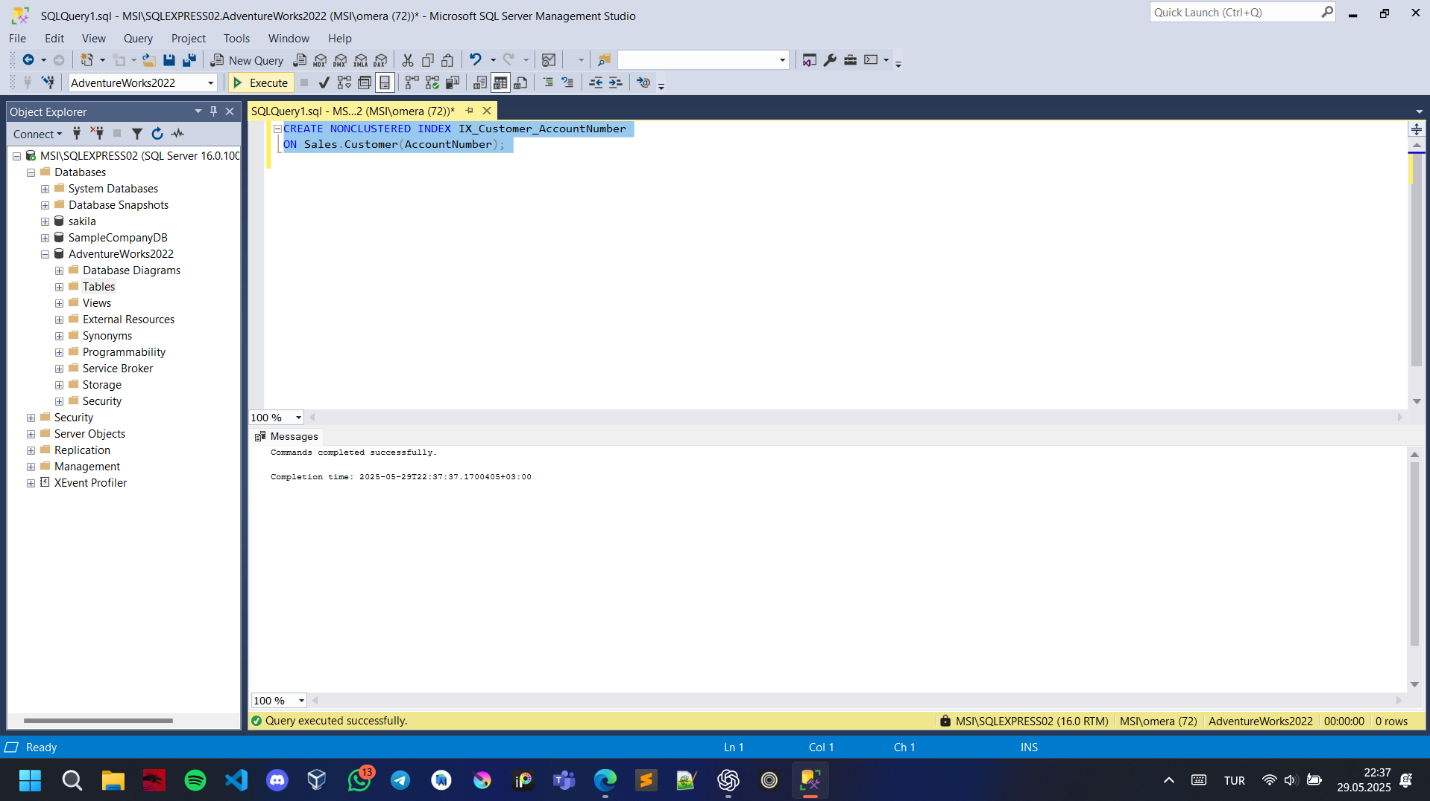
11)



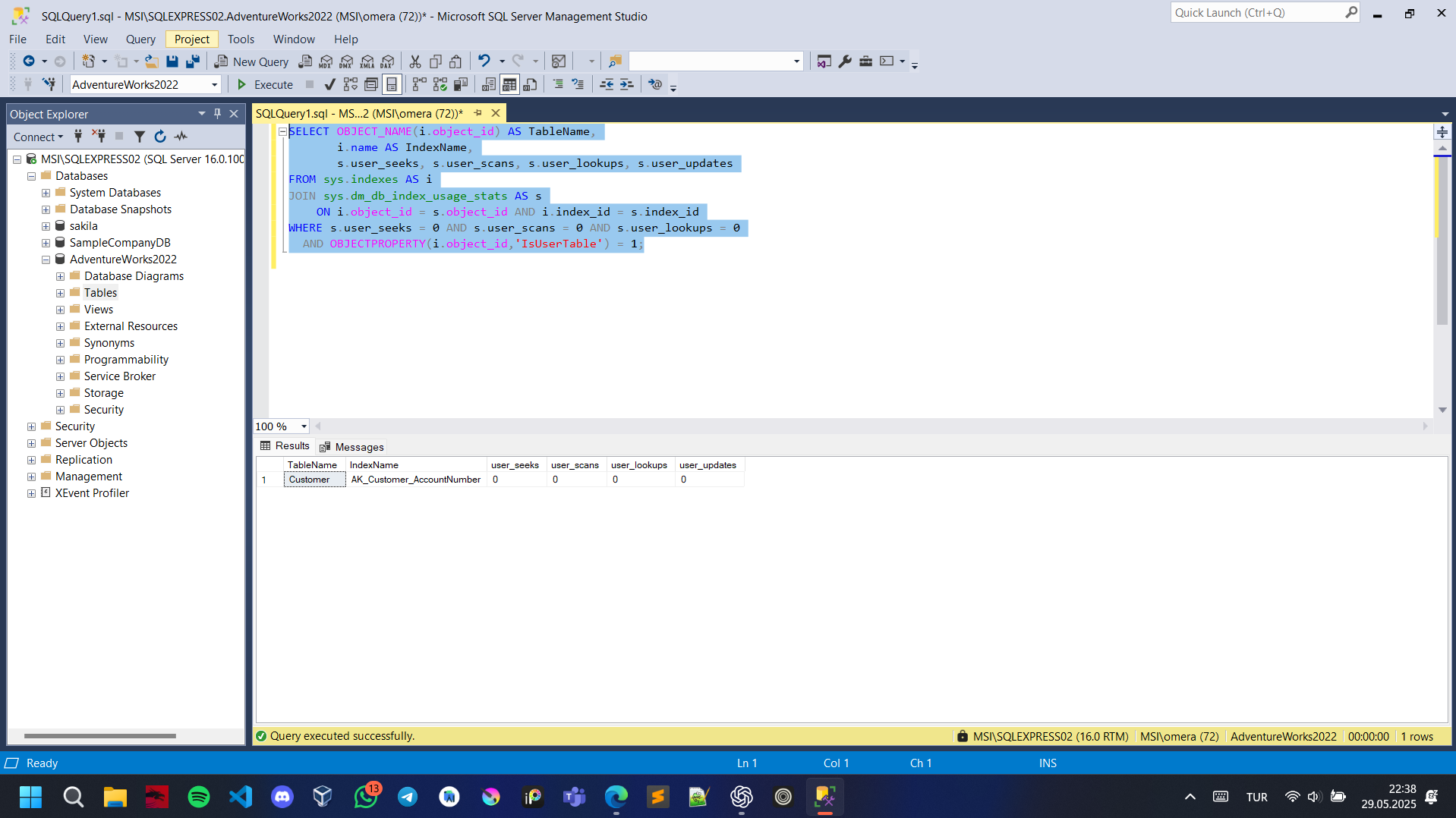
12)



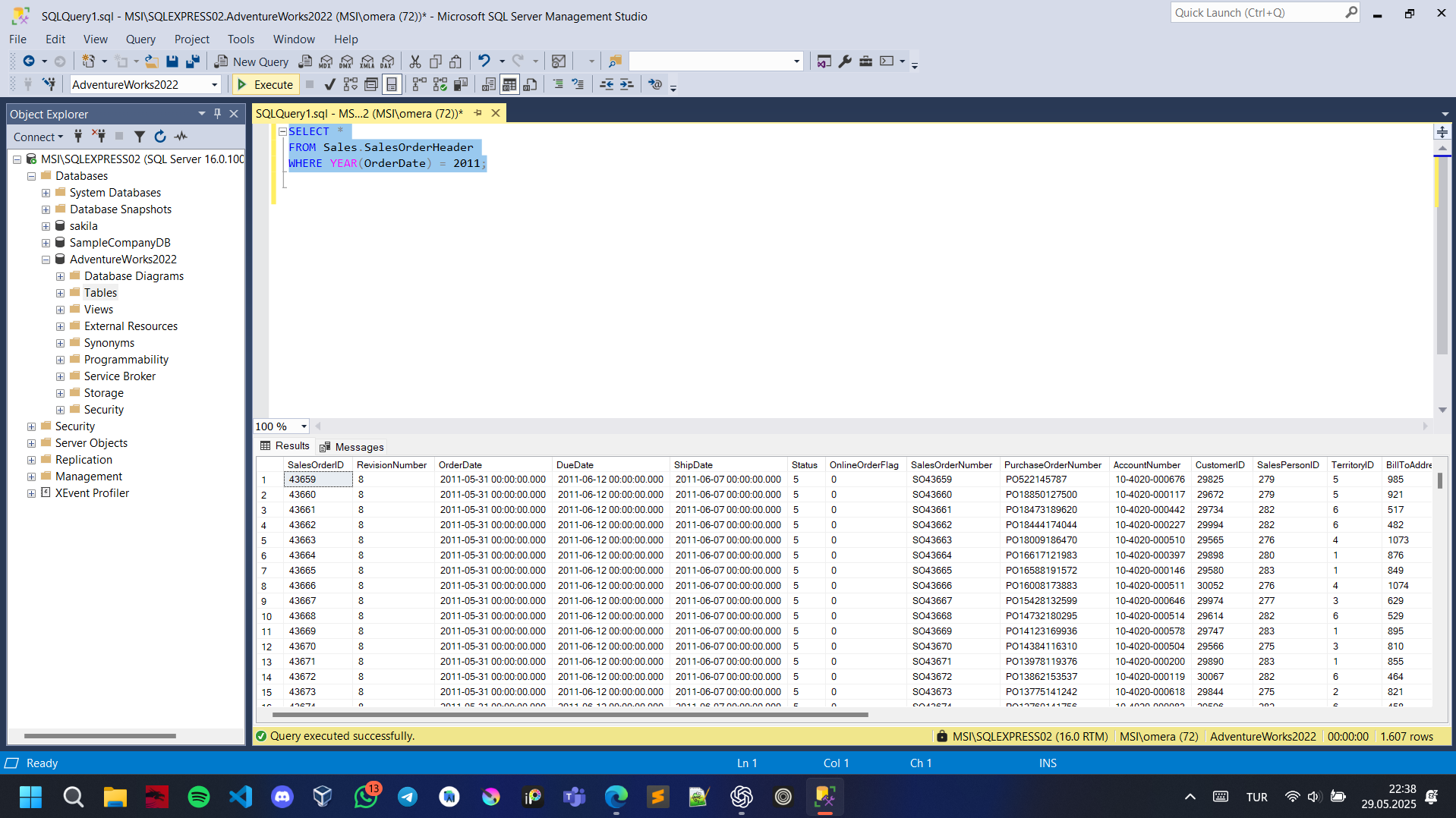
13)



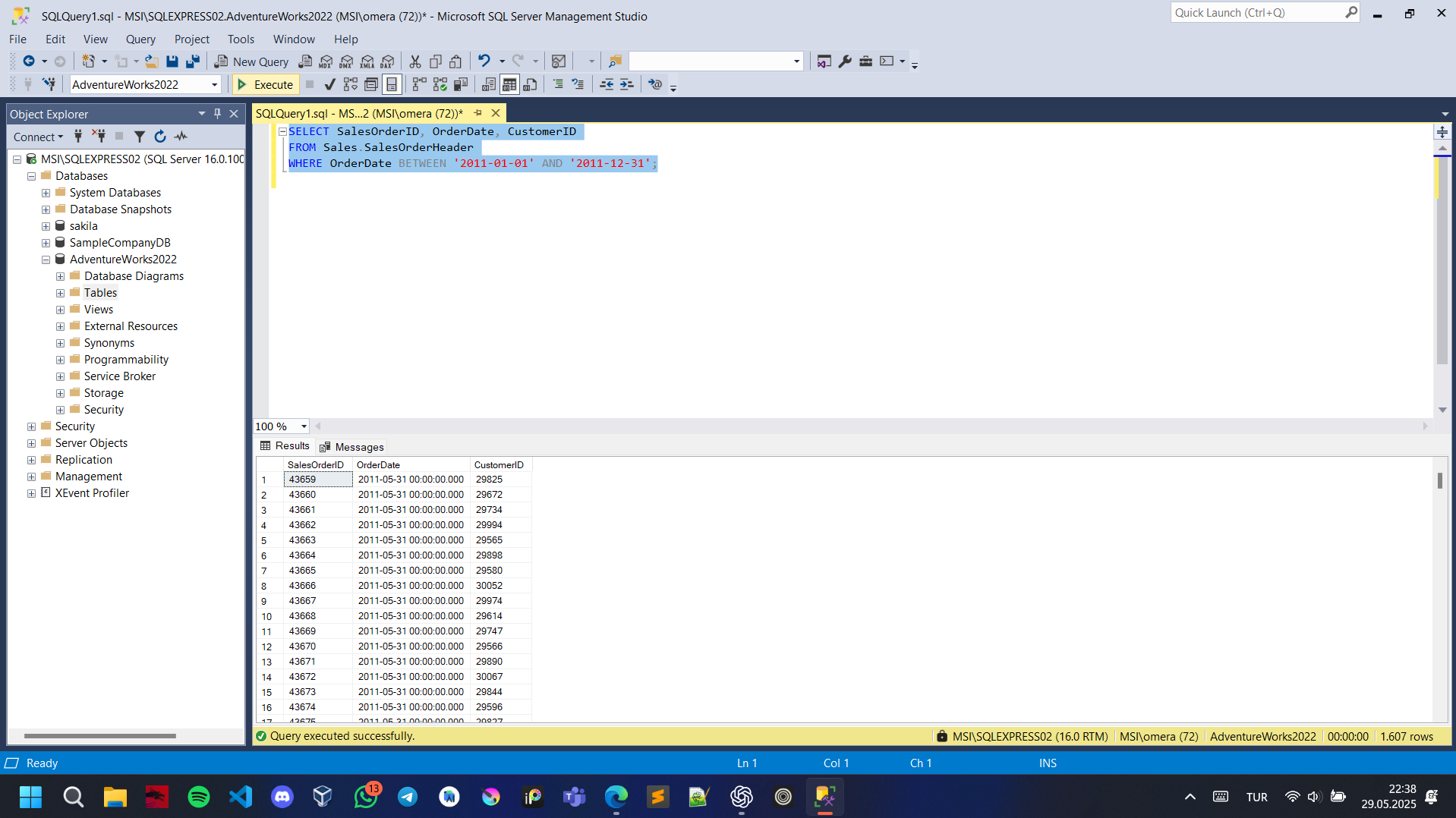
14)



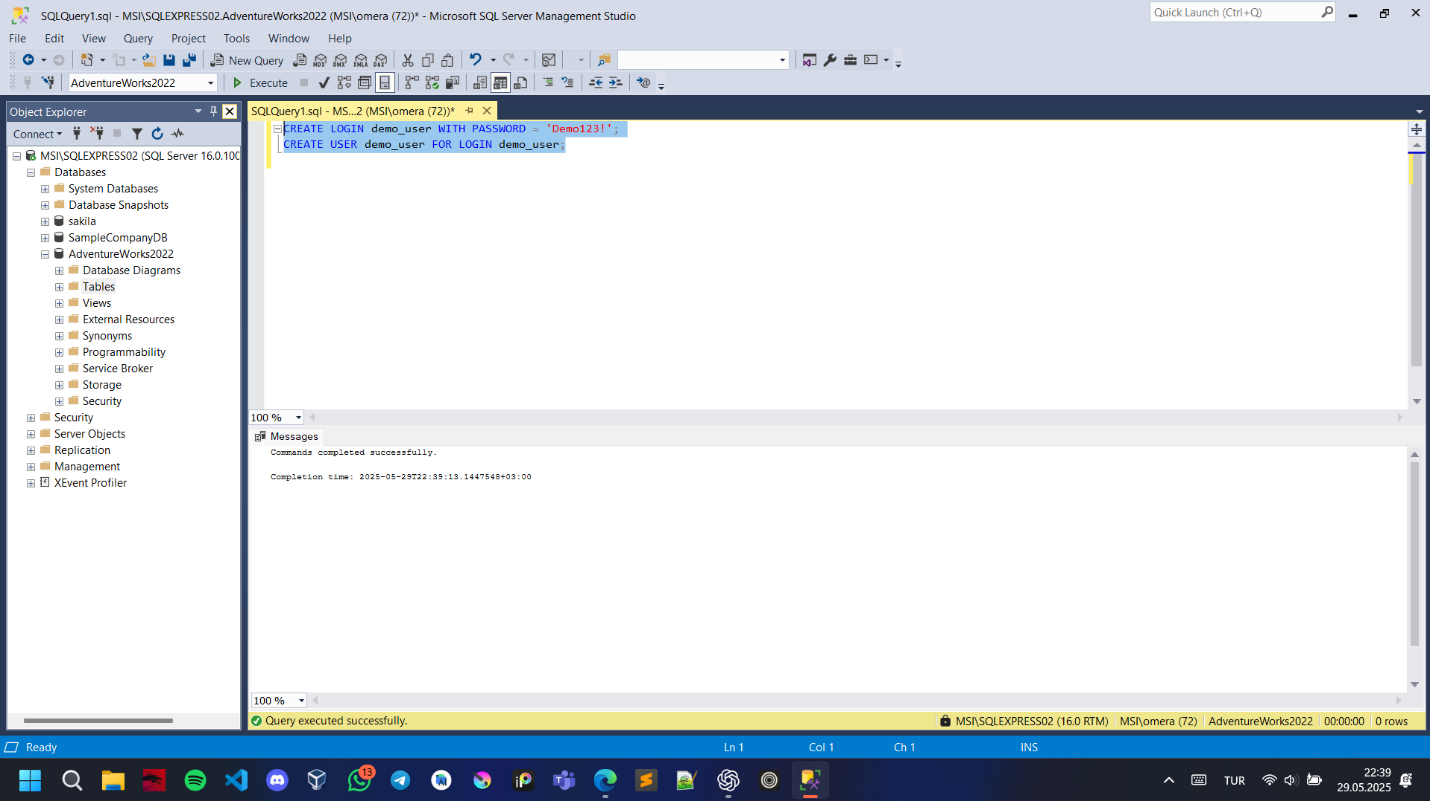
15)



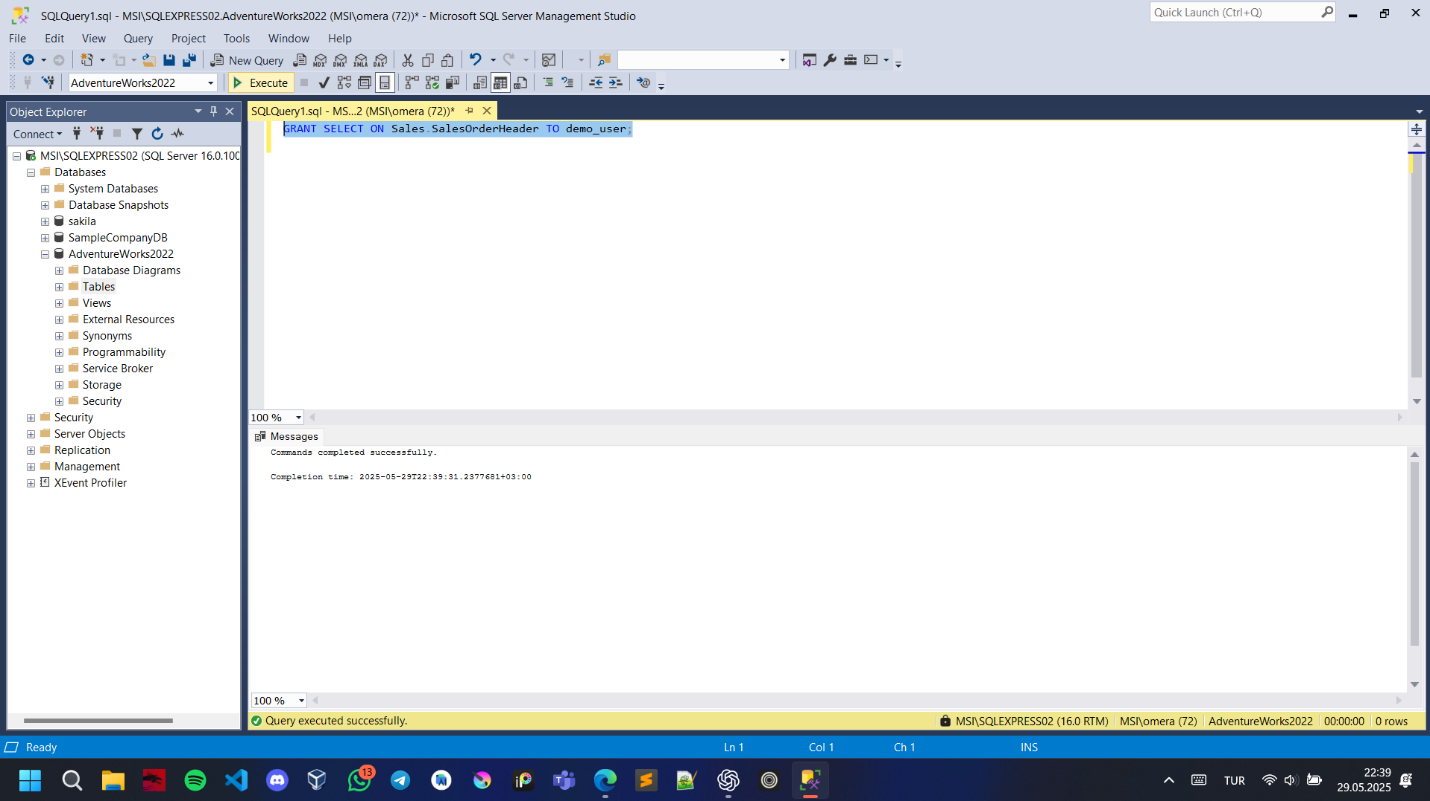
16)



17)



18)



19)

